Protocol for Connection Ethernet Interface to SpaceWire Networks

Evgeni Yablokov, Valentin Rozanov, Alexey Vinogradov
Saint Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russia
{evgeny.yablokov, valentin.rozanov, alexey.vinogradov}@guap.ru

***Abstract*—The article considers a second version of protocol for transmitting and receiving packets via Ethernet interface using the special SpaceWire-Ethernet bridge. The problems of the first version of the protocol were fixed and the new version of the protocol is introduced. Bridge invited to consider in terms of queuing systems. In particular it helps to determine the number of buffers required for processing incoming frames from Ethernet. The article contains the results of calculations and graphics.**

I. INTRODUCTION

Nowadays space industries in Russia, Europe and North America set demanding requirements for on-board communication systems. In accordance with [1], satellites and piloted spacecrafts require networks with high data transmission rates up to tens gigabit per second for data-handling and computer buses and distances of 50 – 100 m between adjacent nodes. Moreover, the acceptable cable mass is about 30 – 60 g/m and galvanic isolation is necessary.

The drawback of SpaceWire (SpW) technology [2] based on IEEE 1355-1995 [3] and ANSI/TIA/EIA-644 [4] standards consists in the deployment of data-strob encoding scheme at the physical layer. Thus, the maximum transmission rate in a SpaceWire link stands at 400 Mbit/s, SpaceWire cable has the recommended length of 10 m and comprises four twisted pairs (with whole acceptable mass of 80 g/m).

Due to SpaceWire cable length restriction (10 m) and necessity of connecting the remote segments of SpaceWire network that is able to transfer packets the new solution is needed. The main condition is minimum delay and maximum simplicity. Gigabit SpaceWire protocol is solving this problems completelyHowever in real equipment of spacecraft and satellites Ethernet technology is used. Thus, our bridge helps SpaceWire users connect through Ethernet and Gigabit Ethernet interfaces.

Ethernet is a protocol that controls the way data is transmitted over a local area network. An Ethernet LAN typically uses coaxial cable or special grades of twisted pair wires. The most commonly installed Ethernet systems provide transmission speeds up to 10 Mbps. Devices are connected to the cable and compete for access using a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. Now only fast Ethernet and Gigabit Ethernet technologies are widely used in most LAN networks. The speed rates of fast Ethernet and gigabit Ethernet are correspondingly 100Mbits/sec and 1000 Mbits/sec.

In order to enhance SpaceWire link characteristics this paper defines special bridge Spw-Gigabit Ethernet. The bridge can be either absolutely transparent for the SpaceWire network and can connect two SpaceWire networks into one, or can be used to connect SpW network through Ethernet interface to end user.

During subject area analysis the following tasks were allocated:

1) Analysis variants of connection Ethernet-SpaceWire bridge in network.
2) Discover and identity problems, that can occur during transferring SpaceWire packets over Ethernet.

Fig. 1 shows a prototype of Ethernet-SpaceWire bridge that was developed. On the board is situated FPGA Virtex(1) and ALTERA(2), Ethernet connector(4), Gigabit Ethernet transmitter, SpaceWire connectors(3), test lines and indication (7,6).

Virtex chip exchanges data with SpaceWire interface and creates payload for Ethernet frame out of SpaceWire packet and forms SpaceWire packet out of Ethernet frame. ALTERA chip forms Ethernet frames, check and create CRC value, transmit and receive data to and from Ethernet transceiver. Ethernet transceiver allows to work on speed

10/100/1000 Mbits/s and send/receive data to/from Ethernet network[1]. In future both Altera and Virtex chips can be used for developing code for different realizations of code for the bridge.

ISSN 2305-7254

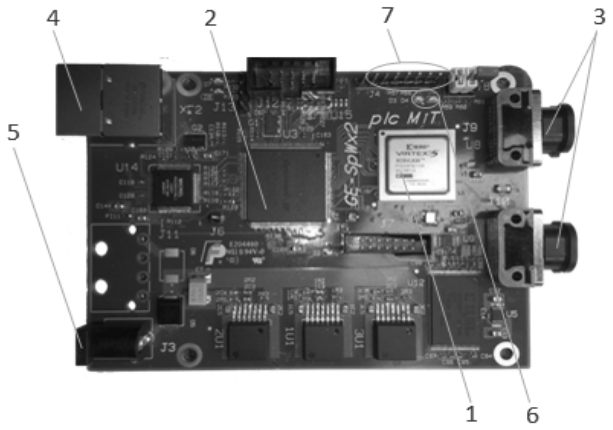


Fig. 1. GigabitEthernet-SpaceWire bridge

Fig. 2 shows block diagram of device working.

FPGA Altera establish connection with network Interface Card of PC or switch and determines speed of the channel. Ethernet frame, comes through Ethernet controller to FPGA Altera where it is processed and CRC is checked and removed from frame. Then frame transmits to FPGA Virtex where it is analyzed, the headers are removed, SpaceWire packet is formed and transmitted to SpaceWire macrocell.

Packets from SpaceWire are analyzed and form the Ethernet frame. Relying on ready-signal from Altera Ethernet frame is transmitted to Altera chip, adding CRC and after is transmitted to Ethernet network.

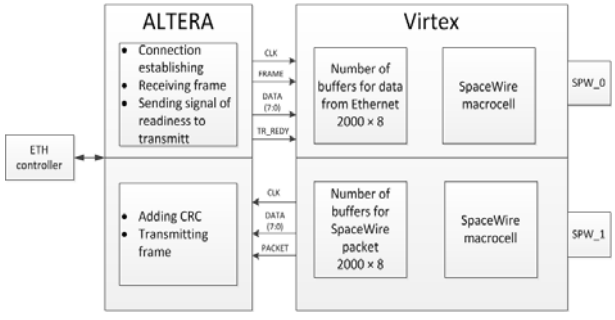


Fig.2. Block diagram of Ethernet-SpaceWire bridge

Different options of bridge connection exists, Fig. 2 shows the three options of connection:

1) Direct connection of the bridge to the PC;
2) Connecting two SpaceWire networks over two bridges with point-to-point connection of the bridge;
3) Connecting the SpaceWire network with onboard network over Ethernet interface (using Ethernet switches for example).

Number one and 2 are implemented by this moment. At the moment, the third connection option is studied separately.

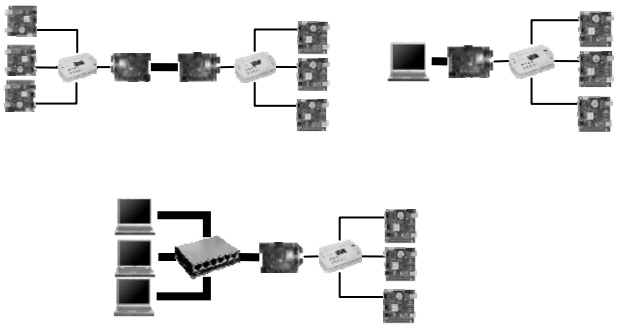


Fig. 3. Variants of using a Ethernet-SpaceWire bridge

II. Discovering problems of Eternet-spacewire transfering

Comparing of the Ethernet frame and SpaceWire packet helps to analysis problems that can appear during transmitting data from one network to another. Formats are similar however there are some significant differences. Ethernet frame has fixed length of header (14 bytes), fixed length of payload (46 to 1500 bytes) and at the end of frame there is CRC (4 bytes) as the end of frame [3]. As opposed to Ethernet SpaceWire may have different length of header depending to address type and information that it contains. Data field may be 1 byte or infinitely long/ The end of the packet is marker that informs about the end of the packet (eop) or the error end of packet (eep) [1,4].

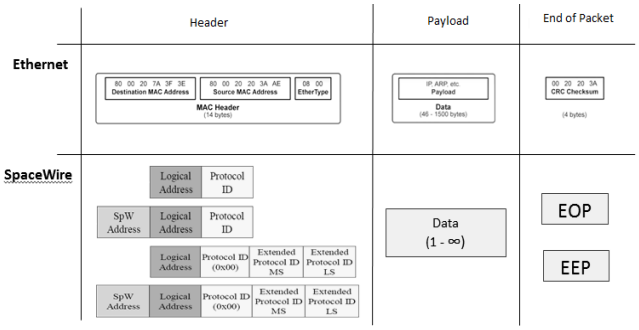


Fig.4. Comparing Ethernet frame and SpaceWire packet

Based on foregoing we can make a conclusion that additional processing of SpaceWire packet is needed before transferring, Especially if the length of the packet is more than 1500 bytes. One of the ways of processing is dividing packets on parts and supplying these data with special informational bytes that can be placed before SpaceWire data and transmitted in payload of Ethernet frame. That bytes will inform about number of part and type (eop/eep/Ccode) to the receiving side.

Fig. 5 is the diagram that shows the relation between useful information and transmitting bytes in Ethernet frame and SpaceWire packet.

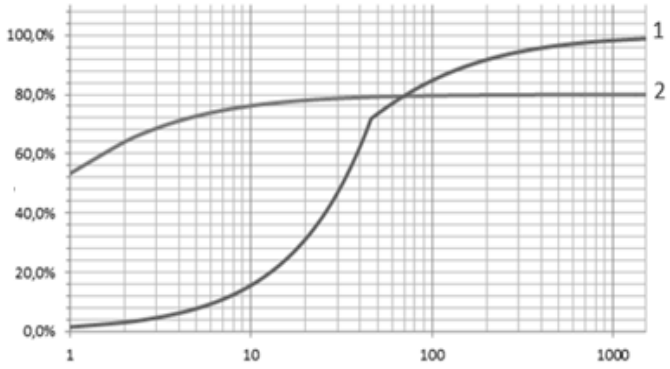


Fig. 5 relation between useful information and transmitting bytes (in percentage) 1 – Ethernet, 2 - SpaceWire

Chart on Fig. 5 shows that maximum of useful data transfers when data field of SpaceWire contain 30-40 bytes. Usefulness of Ethernet frame at this case is low – 50 % of transmitting bytes. Inflection point on the curve of Ethernet is the result of minimum available bytes that may contain frame payload – 46 bytes. So in case of transmitting 1 byte other 45 bytes of payload will be filled with zeros. Maximum useful data is transmitted over Ethernet when payload of frame contains the maximum number of bytes 1500 bytes. It show that bridge developing needs to design transferring rules or protocol, to make this device more productive [11].

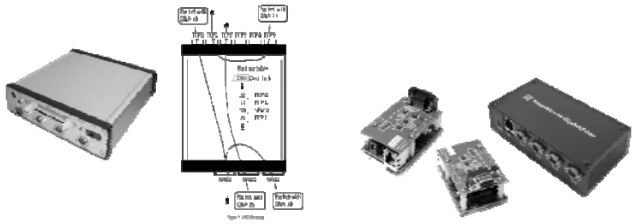
Based on foregoing following problems can be allocated:

1) Low efficiency in transferring short frames.
2) Additional processing of long SpaceWire packets is needed.
3) Low efficiency in transferring CCode by separate frames (see item 1).

III. Comparative Analysis of Similar Devices

Products of Aeroflex company (Fig. 6 a)[6] and ShimafujiElectric were taken to compare with developing device (Fig. 6 b)[7].

The main difference between our block from others - it is a physical realization on the physical and data level of the OSI model, thus making it simple to implement the software protocol. The second difference with the above models – our bridge is absolutely hardware, without any software, making it much faster.



a b

Fig. 6. Ethernet – SpaceWire devices

IV. Development of Protocol

The first version of protocol was described in our previous article. In general two types of the frame packet were used to transfer SpW packets using the frame of Ethernet.

If the Ethernet package contains only a control code – the number of the package doesn't change (Fig. 7).

Ethernet header	Type, number, part number	Length	DATA	CRC

Fig. 7. Spw packet placed to Ethernet frame

The description type of Ethernet frame, transferring the SpW packet, consists of 2 parts – length of packet, type of packet and number of packet (if the length of SpW packet is too big). The first byte of heading contains information of the package type and the package number. All information occupies 1 byte. The type of a package is coded by the first two bits.

01 – the SpW package comes to an end with EOP,

10 – the SpW package comes to an end with EEP,

00 – part of a SpW package (the package isn't finished, there will be a further transfer),

11 – CCODE package.

Number of a package is coded by remained 6 bits. This number is split in two parts – 4 bits, and 2 bits. Number of a package is 4 bit long and is changed cyclically from 0 to 15. Number of a package changes only if the Ethernet package contains data and a package was transferred completely. The last 2 bits code the number of a slice of a package if the package consists of the several parts.

If the Ethernet package contains only a control code – the number of the package doesn't change [6].

The main idea of the first type of protocol was transferring the simple packets through the network, while splitting the big SpW packets into several Ethernet data frame. The CCode was transferred using one Ethernet frame.

Two main problems occurred while developing the protocol:

big packets can't be transferred directly– the frame of Ethernet is restricted to 1500 bytes.

transmission of control codes using one frame in principle is ineffective – 1 byte of data will be transferred by 64 bytes using Ethernet.

The solution, we tried to implement, was to transfer several SpW packets in one Ethernet frame (fig. 8)

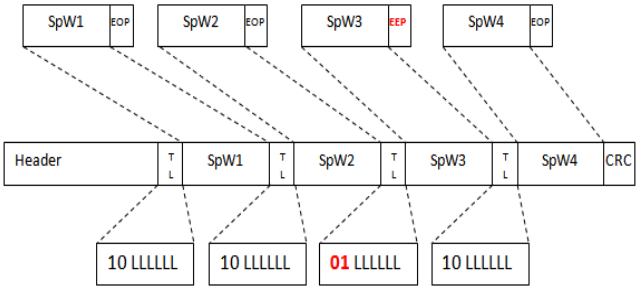


Fig. 8. Several SpW packets placed to Ethernet frame

The SpW packets were divided into parts with maximum length of 64 bytes, with special descriptor, describing the type of the part: data part ending with EOP, data part ending with EEP, intermediate data part (part of packet), CCode. In one Ethernet frame several SpW packets could be transmitted, including CCodes.

The main disadvantages of this protocol are:

1) big delays in transmitting CCode and data frames,
2) several data SpW packets could be lost due to this approach,
3) if the frame lost – there is no info – if the new frame is the part of the lost SpW packet, or the new SpW packet,
4) problems with crediting the buffer place.

Solving the first 3 problems is the main priority of the second version of protocol. Ethernet frame is left untouched, like in first version of protocol, however the information fields are increased.

Descriptor of packet is located in the beginning of the data part of the Ethernet frame. First byte of the descriptor is the type of the packet. Types of packet of the packet are displayed in Table I.

The second part of the descriptor is the informational byte of the current number of frame. The number of frame for each data frame, ccode frame and EFrame are separate, thus all data frames have its own numeration, CCode have its own, and EFrame has also its numeration. The number of frame is going from 0 to 255 in loop.

TABLE I. Types of header

Type	Description	Comments
01h	Full EOF	Full SpW packet(length of SpW packet is less than Ethernet frame), ending with EOP symbol
02h	Full EEF	Full SpW packet (length of SpW packet is less than Ethernet frame), ending with EEP symbol
03h	SOF	The first part of long SpW packet (length of SpW packet is more than Ethernet frame)
04h	MOF	The middle part of long SpW packet (length of SpW packet is more than Ethernet frame)
05h	EOF	The last part of long SpW packet (length of SpW packet is more than Ethernet frame) ending with EOP
06h	EEF	The last part of long SpW packet (length of SpW packet is more than Ethernet frame) ending with EEP
07h	CCode	One or several CCodes are transmitted in frame
08h	EFrame	Optional frame, which have the information of the error or errors occurred on the receiving part.
00h, 09h-FFh		Reserved

The third part of the descriptor is the length of the packet, which consists of two bytes.

The basic Ethernet frame, transmitting SpW data is displayed on Fig. 9.

1	MAC address receiver	MAC адрес sender	Eth Type	Information bytes	SpW Data	CRC
	6 bytes	6 bytes	2 bytes	4 bytes	1-1496 bytes	4 bytes
2	MAC-header – 14 bytes			Data field – 46 – 1500 bytes		

Fig. 9. Ethernet frame with SpW data packet inside.

SpW packet with the length less than the length of the data field Ethernet frame is transferred using one frame, using Full EOF or Full EEF, depending on the type of the end of packet SpW byte. If the SpW packet is more than the length of data field of the Ethernet frame – the packet splits into several part. The first part of SpW packet starts with SOF frame, then (if needed) one or several MOF frames, and the last frame is either EOF or EEF frame, depending on the end of packet SpW byte.

The CCode frame contains one or several SpW CCodes, which can be transferred, using one frame of CCode. How many CCodes can be in one frame and how often must be such frames sent fully depends on the transmitting part,

because some CCodes may be very urgent and some can wait (Fig. 10).

1	MAC address receiver	MAC адрес sender	Eth Type	Information bytes	CCode 1; CCode 2; ... CCode N;	CRC
	6 bytes	6 bytes	2 bytes	4 bytes	1-1496 bytes	4 bytes
2	MAC-header – 14 bytes			Data field – 46 – 1500 bytes		

Fig. 10. CCode frame

Several error situations may occur during the transmittion of the frames directly to the Ethernet network.

1) The incoming SOF, FEEF, FEOF frame has the number 2 more than the previous data frame. The previous data frame was EOF, EEF, FEOF or FEEF.

Reaction:

The event “data frame lost” is generated, number of lost frame (or frames) is saved. The data from the frame received and sent to the SpW network.

2) The incoming SOF, FEEF, FEOF frame has the number 2 more than the previous data frame. The previous data frame was SOF or MOF.

Reaction:

The event “data frame lost” is generated, number of lost frame (or frames) is saved. The EEP symbol is generated and sent to the SpW network. The data from the frame is received and sent to the SpW network.

3) The incoming MOF, EOF, EEF frame has the number 2 more than the previous data frame. The previous data frame was MOF or SOF.

Reaction:

The event “data frame lost” is generated, number of lost frame (or frames) is saved. The EEP symbol is generated and sent to the SpW network. The data from the frame is ignored. Data from the next frames is ignored up to frame EOF or EEF, if the current frame is MOF frame.

4) The incoming MOF, EOF, EEF frame has the number 2 more than the previous data frame. The previous data frame was EOF, EEF, FEOF or FEEF.

Reaction:

The event “data frame lost” is generated, number of lost frame (or frames) is saved. The data from the frame is ignored. Data from the next frames is ignored up to frame EOF or EEF, if the current frame is MOF frame.

5) The incoming CCode frame has the number 2 more than the previous CCode frame.

Reaction:

The event “CCode frame lost” is generated, number of lost frame (or frames) is saved. The CCodes from the frame are sent to the SpW network.

6) The incoming frame has the wrong length or CRC is bad.

The event “Frame lost” is generated. No Data is sent.

The receiving part can inform the other side about errors occurred during the transmittion of the frames. For this purposes the special EFrame may be used. This frame is optional.

The EFrame consist of one or several blocks, each block consist of two bytes. First byte of block contains the type of lost frame (data frame, CCode frame), second byte contains the number of lost frame (Fig. 11).

1	MAC address receiver	MAC адрес sender	Eth Type	Information bytes	Type 1, num 1; Type 2, num 2; ... Type N, num N;	CRC
	6 bytes	6 bytes	2 bytes	4 bytes	1-1496 bytes	4 bytes
2	MAC-header – 14 bytes			Data field – 46 – 1500 bytes		

Fig 11. Error Frame

V. BUFFER SIZE CALCULATION

Our protocol relies on the fact that each Ethernet frame is placed in its own buffer. The buffer must receive the frame of any length. The Ethernet-SpW Bridge is the single-channel system with limited queue [13]. To reduce the amount of the resources used to transfer data without loss of performance it is necessary to calculate the optimum number of buffers for frames[12].

The calculation of performance can be made by either math models or by models of the high-level language such as C++ or SDL. Though the high-level language are more accurate [10], but they takes more time to develop than the math models.

The calculation is performed by means of the package Matlab. For this purpose, based on the formulas located below several graphs are showing the probability of overflow of buffer queue, the number of pending requests with different the number of buffers and the incoming data amount coming per 1 second.

The reduced flow rate can be counted by $\rho = \lambda / \mu$, where λ input intensity, and μ is the output intensity.

The probabilities P0, P1, ..., Pm + 1 are calculated by the formula (1).

$$\left\{ \begin{array}{l} p_0 = \left\{ \begin{array}{l} \dfrac{1-\rho}{1-\rho^{m+2}}, \rho \neq 1 \\ \dfrac{1}{m+2}, \rho = 1 \end{array} \right\} \\ p_k = p^k \bullet p_0, k = 1,2,...,m+1 \end{array} \right\} \quad (1)$$

where p0 - the likelihood that SMO is free and can receive the frame, p1 - probability that SMO is busy, and no requests queued, p2, ..., pm + 1 - probability that SMO is busy and the queue there are 1,2, ..., m requests.

Probability of deny Potk is calculated Potk = pm + 1.

The average number of requests in the queue R is calculated by formula (2).

$$\bar{r} \left\{ \begin{array}{l} \dfrac{\rho^2 \left[1-\rho^m (m+1-m\rho) \right]}{(1-\rho^{m+2})(1-\rho)}, \rho \neq 1 \\ \dfrac{m(m+1)}{2(m+2)}, \rho = 1 \end{array} \right\} \quad (2)$$

The result graphs are shown on Fig. 12, 13 and 14. The coefficient of maximum load of The Ethernet was taken from 20% to 100%. Number of buffers are more than 10, and less than 100.

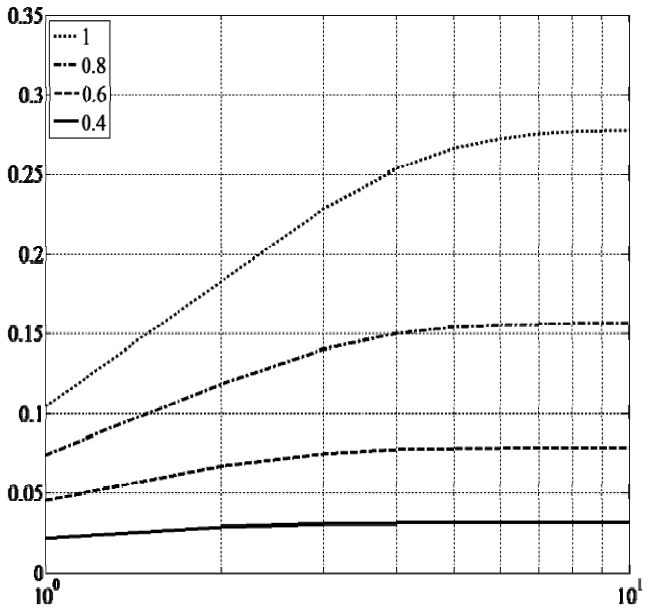


Fig. 12. Dependence of the average number of requests in the queue on the number of buffers

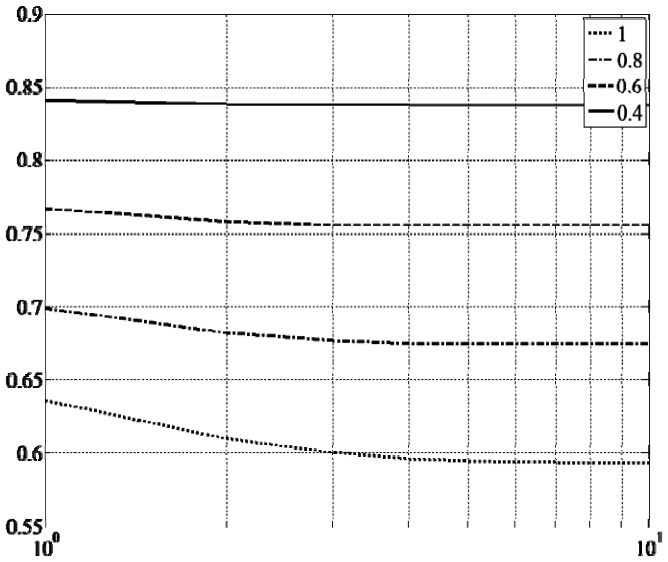


Fig. 13. Probability that system is ready for receiving the new Frame

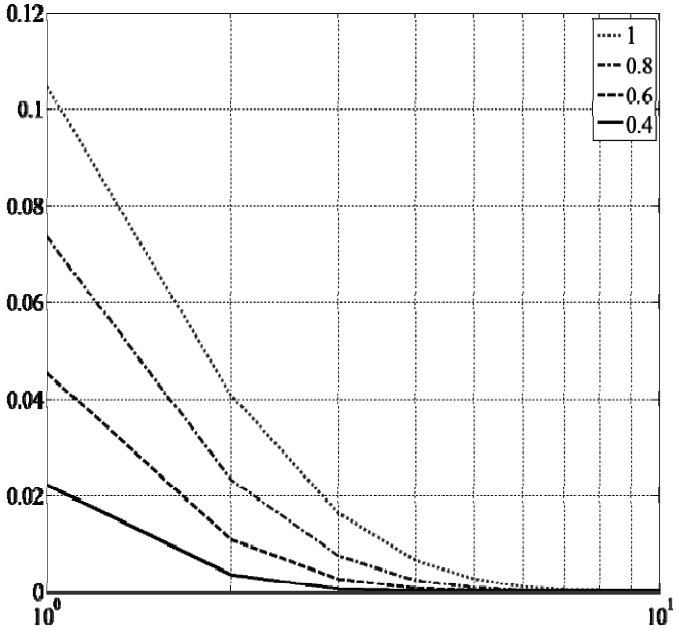


Fig. 14. Probability of deny of the request depending on the number of buffers

The SpW speed is lower than Ethernet, and if the load is rising – more free buffers are needed.

VI. CREDIT EXCHANGE

In most of cases the buffer number are limited by the chip [9], for example – in FPGA Spartan6 it is not possible to create more than 10 buffers for Ethernet frames. In such cases the exchanging of credit (number of free buffers) are needed. For this purpose all frames have the new information byte, which have the number of buffers free for receiving the Ethernet frame.

In the beginning of the exchange the transmitter suggests that the receiving part has at least two buffers free. After the receiving the new info about crediting the transmitter update its own counter of frames it can send. If the transmitter has no data to send he must sent the data frame with 0 length, with only credit information. Every credit means that one frame can be received.

Transmitter has the one counter of the frames it can transmit, which comes from the other side (transmit counter). The second counter contains the info about the number of free buffers of the receiver (receive counter). This number transmitter send in every frame going to the other side.

The process of receiving is simple: frame is placed to next free buffer, receive counter is decreased by one, after processing the receive counter is increased by one.

To avoid a situation when the frame was received after the credit information was just sent (the credit is already used by the coming frame) one of the proposed strategies must be used:

1) Always send the number of credit less than free buffers. This is not recommended, because in this case not all the buffers are used, thus the number of buffers must be significant.
2) Create a special delay counter, which will take into account that the other side has considered the received package.

Every time a node sends a packet to the network it increases the value of the delay counter on the base value. Delay counter reduces its value by 1 using the local frequency signal. If the package comes from the opposite side when the counter value is not 0, the counter value is saved. After analyzing the value of the credit field of packet, the saved number is also analyzed - if it is less than the base value, but greater than 0, the transmit counter is set to 1 credit less than the value from the received frame. If the value of the counter is more than base value, but less than the base value multiply 2 - lending counter is set to 2 lower than the value from the packet, etc.

Base value depends on cable length and data rate. At frequency of 1.0625 GHz and the maximum cable length of 100 meters the initial counter value should be 80. If the frequency is increased up to 2.5 GHz - counter value must be equal to 200.

VII. Conclusion

As the result of the work the following conclusions were made:

1) To implement a new version of protocol, presented in this paper, on the prototype device, we need to use 4 buffers in receive module of Ethernet part. It gives maximum speed to receive frames from Ethernet interface and allows to catch all frames and process them by turns.
2) New version of protocol enables crediting function, so we can be sure that all frames (and all data) will be received and nothing will be lost.
3) Besides that basic system of errors handling will be implemented, based on header in Ethernet frame containing the information about transmitting SpaceWire packet
4) Increasing the number of types of Ethernet frames containing SpW Packets allowed us to pinpoint errors in the transmittion and also inform the other side about these errors. In the future, this may allow to implement a guaranteed data delivery.
5) The latency of transmitting was reduced in the new protocol. This is achieved by creating the CCode frame, which can transmit several CCodes, thus greatly improving efficiency.

Our prototype bridge is working on two low levels of the OSI model, thus making it simpler than all other devices, using protocols of the higher levels.

References

[1] Yuriy Sheynin, Felix Shutenko, Elena Suvorova and Evgenej Yablokov, High-rate serial interconnections for embedded and distributed systems with power and resource constraints, Proc. SPIE 6983, 69830I (2008); doi:10.1117/12.786883
[2] MicrelInc, Gigabit Ethernet Transceiver with RGMII Support, Reversion 1.2, February, 2014
[3] Ethernet Standart IEEE 802.3
[4] Yu Scheinin, T.Solohina, Ya.Petrichkovich, SapceWire technology for parallel systems and on-board distributed systems, ELECTRONICS: Science, Technology, Business 5/2006
[5] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008
[6] AerofexGaisler, GRESB Ethernet/SpaceWire Bridge, Version 1.5.9, January 2014
[7] SHIMAFUJI Electric Inc., Document No. EN210439003-0, STAND-ALONE SpaceWire to Gigabit Ether H/W specification, Date Published July. , 2012
[8] Virtex-5 FPGA Select IO Resources User Guide
[9] S. Balandin and M. Gillet, “Embedded Network in Mobile Devices”, International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Issue 1(1), pp 22-36. 2010.
[10] S. Balandin, M. Gillet, I. Lavrovskaya, V. Olenev, A. Rabin, A Stepanov,“Co-Modeling of Embedded Networks Using SystemC and SDL”, International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Vol. 2 (1), pp 24-49. January-March 2011.
[11] Jin, H.B., Zhong, C.Q. An optimal algorithm for real-time Ethernet buffer queue based on queuing theory, Volume 52, Issue 1, January 2012, Pages 95-99.
[12] Marcin Bienkowski, An Optimal Lower Bound for Buffer Management in Multi-Queue Switches, arXiv:1007.1535
[13] Joan Vila-Carbó, Analysis of switched Ethernet for real-time transmission , Joaquim Tur-Massanet and Enrique Hernández-Orallo Universidad Politécnica de Valenci.